NPS-CS-09-003

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

METHODS FOR CREATING REALISTIC DISK IMAGES FOR FORENSICS TOOL TESTING AND EDUCATION

by

Loren E. Peitso
Simson L. Garfinkel

March 17, 2009

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Daniel T. Oliver
President

Leonard A. Ferrari
Executive Vice President and Provost

Approved for public release; distribution is unlimited

This report was prepared by:

Loren E. Peitso
Senior Lecturer and Associate Chair for Administration

Simson L. Garfinkel
Associate Professor

Reviewed by:

Released by:

Peter Denning
Department of Computer Science

Karl Van Bibber
Vice President and Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
17–3–2009	Technical Report	2008-11-01—2009-01-30

4. TITLE AND SUBTITLE

Methods for Creating Realistic Disk Images for Forensics Tool Testing and Education

5a. CONTRACT NUMBER

M92367

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Loren E. Peitso, Simson L. Garfinkel

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-CS-09-003

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NIST

10. SPONSOR/MONITOR'S ACRONYM(S)

CFTT

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this report are those of the authors and do not necessarily reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

Both testing of computer storage forensics tools, and education in conducting computer forensics require reference drive images with known characteristics. Without a known ground-truth it is not possible to fully verify the ability of a tool or a student's analytical technique on whether they capture the important data residing on the drive. Due to privacy concerns existing corpa of drive images from real users cannot be used, so we must construct drive images that do not contain any privacy-related information. This paper discusses methods to generate drive images constructively and the concerns that must be taken into account to ensure they are realistic, reflecting not only the particular testing scenario desired, but also appropriate "background noise". Further we discuss competing methods to accomplish this and propose a means of automating the entire process.

15. SUBJECT TERMS

NPS Technical Report

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Simson L. Garfinkel
Unclassified	Unclassified	Unclassified	UU	35	19b. TELEPHONE NUMBER *(include area code)* 831-656-7602

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

1 Introduction

Computer Forensics is unique in computer science in that the data that is of interest to forensic practitioners is frequently highly personal information that the subjects of a legal investigation are attempting to conceal or destroy. Such information typically includes chat logs, photographs, office document files, and email messages. This information may have been deleted by the computer user, may have been partially overwritten, or may have been encrypted. A significant part of the educational process involves teaching students how to find this data, how to recognize it when they have found it, and how to assemble many different data elements into a single investigative narrative that explains *what was done* and *who did it*.

Both the testing of computer forensics tools and the teaching of computer forensics in a classroom environment requires forensic data that is representative of data that will be found in the field. We call such data *realistic*. But from where should experimenters and educators get such data? Although it is common practice for researchers to use their own hard drives when they are first learning to use the tools, such highly personal data is not appropriate for either a formalized testing or an educational environment. Another common approach is to purchase used hard drives on the secondary market, since such drives frequently contain data from their previous users [3]. Unfortunately, neither of these approaches are appropriate either for tool testing or for education. In both of the environments, researchers, educators and students need data that is both rich in detail and that is free of any personal information, so that results of tools running on the data can be freely published, shared with tool developers, or made the subject of classroom demonstrations.

This paper discusses approaches for creating realistic disk images that are free from confidential information. A key goal of this paper is that the created disk images should be known all the way down to the sector level. We present an approach that does this, by constructively generating disk images and using the knowledge of every file written to the image to maintain ground truth-all the way down to the sector level.

1.1 Areas of Focus

The overall process of generating a realistic disk image has two major areas of focus. Document-centric issues and filesystem specific concerns. These are both trivial to handle by hiring a number of human actors to role-play and work on a computer for several months or even years, and let the normal interactions build a disk image while simultaneously logging every operation

that caused a file to be saved or modified. Obviously that is not a practical solution, both from an expense as well as a time-to-create point of view. So we cover document and filesystem creation with an eye towards automation from the start. Some specific sub-tasks are sill most effectively handled by human interaction, but the majority the process revolves around the goal of simulating a personal computer system drive which today can tally well over one million individual files. Scaling creation methods in this problem space require a substantial amount of automation to get to that level in a reasonable timeframe.

Document-centric concerns cover those issues which must be handled to generate privacy-free documents and document metadata. The document metadata is an important signifier of a documents relationship to the other documents on the drive and any scenarios embedded within the overall set of documents on the drive. We also cover the tradeoffs of using various methods to create content for these documents, many of which can generate text that is convincing at a passing glance, but not coherent in the whole. While these filler-text methods are not appropriate for files which are central to an embedded scenario(s) on the drive, they are very useful for those documents which are on the drive but peripheral to the central scenario(s), the "background noise" through which the interesting information "signal" must be discovered.

The filesystem-centric concerns deal with metadata and wear issues as well as methods which should allow us to create disk images that exhibit realistic characteristics. This encompasses both filesystem directory accessible files and orphaned sectors no longer part of the filesystem tree. We cover consistency between document metadata and filesystem metadata and the challenges time creates when we generate a constructive disk image on a compressed timeline, compared to the timeline of how a real user would affect the drive. We also cover issues dealing with the generation of orphaned sectors for a constructive drive image. Finally we discuss the tradeoffs of various means of writing a filesystem to a constructive drive and then propose a hybridized method which can take advantage of the organizational layout of a real drive image, but write only constructive data to the target filesystem.

2 Documents

This section is about creating documents prior to the documents being inserted into the filesystem. Some filesystem related concerns are addressed, such as will the document yield a set of unique sectors once written by the filesystem, but these topics are addressed by formation of the document content itself rather than involving both document creation and filesystem incorporation simultaneously. The primary concern at this stage is to make believable, unique documents

that will generate a set of unique MD5 sector hashes once written to the drive by the filesystem. Believable, unique documents refers to the property that the documents are human readable and decipherable. These documents will not be guaranteed to be content complete, grammatically perfect, or semantically consistent, but a person reading them will recognize them as a sentence-based document as opposed to random streams of characters.

One of the primary difficulties with using existing hard drive images as a source for redacted drive images is the embedded privacy-related information in the free form content of the files. The difficulty in *guaranteeing* that all obviously direct privacy-related information has been reacted from *all* free form text is currently beyond the state of the art. It is a problem made even more difficult by indirect or inferred privacy-related content. The risks to individuals and programs simply does not allow us to currently use redaction routines to scrub existing free-form content and be comfortable there are no privacy-related violations in the remaining image content. Privacy-related redaction in document and filesystem metadata is a much more controlled problem that is believed to be achievable and will be covered in Sections 2.1.1 and 3.3.3 respectively.

So this leaves us with three options, use an existing image redacting the filesystem metadata and replacing the free form text, generating metadata from scratch independently from writing document files, or allowing an application to generate the metadata during normal in-application file modification operations. All three methods can create a drive image populated with synthetic content that is privacy content free. There are still tradeoffs though, a generated image may tend to look artificially generated unless care is spent in manipulating the metadata to simulate the passage of time effectively. Redacted metadata image substitution will face challenges in preserving conversational or transactionally related threads within content of document subsets unless human generation is used or advanced conversational software agents are developed. Generation schemes can plainly create related document content to simulate conversational threads, but will need very careful manipulation of document and filesystem metadata to maintain the threads proper context. These issues will be explored more deeply in the following two subsections, Section 2.1 covering document-embedded application-specific metadata and Section 2.2 covering free form content creation.

A slightly higher level consideration in creating a drive image are the metadata "scenario" set the drive image tells over time. Some drives such as flash cards for digital cameras have negligible scenarios, the camera is the lone user and the varying component is the writing and

deletion of images. Drives from personal computers used on a daily basis over several years may have many complex scenarios that reflect the primary user, guest users, shared documents, application and operating system updates just to name a few. The set of scenarios emerges from the time-series of the metadata, changes seen, changes expected but unseen and patterns or clusters of other-user generated metadata appearing on the drive. A realistic drive image should be generated with at least this set of considerations driving document creation, modification and saving to embed a consistent scenario set in both metadata and content on the drive. The ability to create such consistent scenario sets currently only exists with real users, but we believe it is a reachable goal to produce drive images with believable scenario sets embedded in the metadata of the drive via automated methods. We also believe automation should be able to provide content which is realistic as far as forensic discovery is concerned, but when examined closely by a person is obviously not generated by a live human in-context for the document the content is contained within. This standard should be sufficient for all but the rive scenario's critical path files.

2.1 Document Metadata

Document metadata is application specific data that describes the document but is not part of the document content. It may contain simple statistics such as word count, historical data such as creation data, or complex revision management tracking information. Significant user-related information can be gleaned from documents with complex metadata, so maintaining the availability of document metadata is desirable for constructive drive images. Whichever of the three previously mentioned methods: redaction; from scratch generation; or within application generation; are used to manipulate document metadata, there are a number of considerations which remain common and are covered in the following paragraphs.

Per-application creation and application version-change artifacts in long-term images. Applications append or embed metadata to a document at specific programmatic triggers that remain consistent throughout the lifetime of a specific application version. Typical metadata inclusion triggers might be application creation or post-modification saving. Once the metadata inclusion triggers are understood for an application version, the effects those triggers will have can be analyzed against a existing drive image or reflected in how they get applied during creation of a constructive drive image. Specific considerations are discussed later in Sections 2.1.1 thru 2.1.2.

Application version changes may alter the metadata fields and/or inclusion triggers, not only for new documents compared to documents created with earlier versions, but the new version may overwrite portions of metadata with the new schema. It may even be possible that programming bugs cause updates to document metadata without removing previously included metadata. This may provide highly interesting forensic data where the user is unaware that previously existing metadata is still embedded in their document(s). These situations make it valuable to precisely know the metadata behaviors of applications on a per-version basis, especially any user-unexpected behaviors. Without knowledge of what an application "should" put in application metadata it may be difficult to discover a scenario or verify one as told by the existing documents. On the other hand, it would also be much more difficult to generate drive scenarios in a realistic, or at least believable, manner if the rules for what metadata should be where are missing or incomplete. Cataloging the metadata generation behaviors of known applications on a by-version basis provides a solid background to work from in a comparative manner. Knowing these behaviors allows drive images to be created which exhibit the same variations over time, including incorporation of results from any known odd behaviors.

Completeness or Incompleteness and Correctness or Incorrectness. Just because document specific metadata is available for the application which created the document originally, it does not mean the information contained within the metadata will be complete or correct. For example, the *User Information* preferences in Microsoft Word are the source for the user metadata when Word creates and/or manipulates a file. Should a user not fully fill in the *User Information* preference pane some of the metadata fields will not be populated. It is also possible that the data sources for metadata may be incorrect due to innocuous information like typographical errors, or if a computer changes primary users without updating the metadata source information. This may actually provide interesting information if at some point in the lifetime of a drive image the completeness or correctness of the metadata were to change in a time identifiable pattern. It is also be possible that a user may intentionally attempt to remove, manipulate or obfuscate the document metadata for legitimate reasons or otherwise.

Another aspect to correctness or incorrectness of document metadata is inclusion of hidden document content. Some applications such as Microsoft Word have a *track changes* function which maintains multiple versions of the document via a revision history maintained in metadata. The revision history normally contains both who modified the text and what those modifications were. It is possible that a user may not be aware that hidden revision tracking metadata is in the

file and leave forensically interesting information they might not have desired to have included in the file. The ability to positively control and manipulate the completeness and correctness of document metadata, including revision and/or other hidden content metadata, enables ground truth to be established for images constructed to containing files with these types of metadata relationships.

User Metadata Consistency Across a Drive. Related to the previous item is user metadata consistency across the drive image. Not just within documents created by a single application, but commonalities and differences in a cross-application manner. Many reasons can be responsible for differences in user metadata across the drive. Some of these differences are innocuous and result in variations of semantically consistent data, for example application 'A' places the user name John Doe in it's document metadata, but application 'B' inserts JohnDoe. Other differences may indicate documents created by another entity on a different computer, or changes in the user environment, such as older documents having one postal address and newer documents having a different postal address. Simulating these user metadata commonalities and differences will provide a much richer set of test cases for forensics tools and students.

Variations From Removable Media Being Used in More Than One Device. A special case of the above is removable media. Removable media by its very nature is normally used across multiple machines, and the primary user's metadata can only be as consistent as network policy and user attention to detail dictate. Removable devices used on networks with strong version control and roaming profiles will likely exhibit fewer user metadata differences, than devices that are used across multiple networks or external computers.

Shared Document Metadata. Shared document metadata can exhibit variations due to any number of the above considerations. Inclusion of documents with other-user metadata on a drive image does not automatically imply a direct, overt connection between the document creators. The ability to download documents over the internet makes the analysis of connections between various users/authors much more difficult to positively ascertain. Drive image creation should therefore carefully balance the mix and connections of shared documents placed on a constructive drive image so as to create desired user relationships and some level of distracting background noise without inadvertently creating unintended red-herring user relationships.

2.1.1 Document Metadata Redaction by Substitution

Using document metadata redaction on files from an existing drive image provides the pattern for the background scenario(s) a dive image tells, and does so in a demonstrably realistic manner. The existing source-drive scenario set is already real, not just realistic, captured from a drive image from an existing drive image corpus. Thus, when the source metadata on the drive is replaced field-by-field with constructive metadata the structure and organization of the constructive metadata is substantially similar to that of the source drive, honoring the actual metadata wear, variations and cross-connections inherent in the source drive's metadata scenario set. One beneficial aspect that stands out is the property of faithfully maintaining timing-related metadata. Metadata creation or generation by other methods handled in a batch or accelerated manner requires the management of timing-related metadata to avoid artificialities and errors. Redaction of privacy-related data need not artificially modify timing related metadata, as time stamps and version change artifacts can be explicitly preserved. This is beneficial to the realism and believability of the overall drive image metadata scenario set.

Implementation of a metadata redaction scheme appears straight-forward although not without significant effort to produce a robust automated process. Metadata, being structured data within a fairly narrow per document-type context, generally lends itself to direct methods for redaction. This is because the semantic information for each field is not difficult to ascertain and it remains consistent on at least an application version-by-version basis. Thus fields which are expected to contain personally identifiable information can be examined and scrubbed if any data appears at all. It is even possible to maintain a data structure which tracks redacted data conforming to patterns such as names, initials, addresses, phone numbers etc., and the constructive data used to supplant it. Then as those same source data items are redacted in other document's metadata, the same substitutions can be made preserving those forms of cross-document relationships. The less constrained situation of document revision metadata can be handled with a combination of the above substitutions and document content substitution as discussed in Section 2.2 for free-form text. Creating a drive image this way reduces opportunities to generate a target image which could be considered unrealistic or unfair. On the other hand this methodology on it's own does not create drive images which have a guaranteed set of "interesting" metadata scenarios to test tools and students against.

2.1.2 Document Metadata Creation from Scratch

Creating document metadata from scratch guarantees that there will be no privacy-related information contained in the final drive image's document metadata, but the process of creating metadata this way is far more involved compared to the redaction method. A critical step to generating realistic document metadata is determining and cataloging the metadata generation behaviors of the applications that are responsible for generating the documents on the drive image. Without the appropriate knowledge of what applications produce what metadata when, generating document metadata becomes an approximation that can be less than realistic. The risks of forensic test drive images containing approximate rather than realistic document metadata is that forensic application designers will be rewarded for skewing their applications to perform better on the faulty test images, which may sub-optimize and/or overstate their operation on real disks. Conversely, the complete control of document metadata inherent in the generation from scratch method provides great flexibility to tailor document metadata to generate a drive scenario set or singular scenario.

Metadata generators need to be configurable to allow matching application specific properties as well as other properties such as the propensities for metadata to be partial, incorrect or missing altogether. For example, not all users of Microsoft Word fill in their preferences for User Information which supplies their document metadata, or they may have originally filled it in but changed job positions or mailing addresses. To properly implement configurability such as this, once an applications per-version metadata creation behaviors are known, we need to move to the next level of existing image analysis to determine what constitutes realism and how to quantify it so that it may be generated into a constructive drive image.

As noted earlier in Section 2.1, many factors external to the application that generate or maintain a file affect the final metadata embedded in a particular document. Choosing which of the possible factors to apply needs to be done in a manner that yields a realistic result. Statistical analysis of a drive or number of drives can provide guidance in this choice. Probability distributions can be determined for the various metadata factors which can then be used to randomize choices for metadata generation and modification which will allow us to create generated document metadata that exhibits the same statistical profile as the reference drive(s). Further research is needed to determine how fine-grained the statistical profiling must be to enable generation of realistic drive images.

We anticipate that metadata generation from scratch can be a useful tool for creating specific

drive scenarios to embed within constructive drive images. Creation of tailored images for education and/or training for drive analysis can be streamlined, allowing the instructor to embed the specific target scenario into an existing target image which shows signs of long term use and provides sufficient "background noise" to keep the forensic work challenging and not artificially simplistic.

2.1.3 Within Application Document Metadata Creation

Generation of document metadata by directly running the application in its normal running mode to create, save and maintain documents is the final method of metadata creation discussed. This is a completely privacy-safe method for generating constructive metadata which also guarantees the correctness of the application specific metadata. The basic procedure to do so is fairly straightforward for a human operator: run the application to create, save and/or modify documents. This manipulation and creation of new documents with the associated metadata by a human operator can be quite slow, and if a large number of documents are required, monotonous. Automated application control can be accomplished to relieve monotony and increase accuracy, specific methods for doing so are discussed in Section 2.3.

2.2 Document Content

Document contents are the relatively free-form text and graphics which makeup the human readable document itself. Since there is negligible constraint on what this content may be, the problems in generating believable, realistic, content supporting specific scenarios and/or scenario sets are far more difficult than the problems in generating realistic metadata. There are also substantial challenges in synchronizing content with metadata as well as maintaining consistent relationships within the drive scenario set. Section 2.2.1 will cover redaction of existing documents and the difficulties therein. We anticipate the difficulties in direct privacy-related information redaction will lead to using combinations of techniques from Sections 2.2.1's Document Content Redaction by Complete Substitution and 2.2.2 Document Content Creation from Scratch.

Until software agents exhibit enough flexibility to generate believable and realistic content it is proposed that automated content creation be used to provide the voluminous background noise inherent in the vast number of files on a drive, and when necessary provide scenario specific content created by a human. Using a technique such as this relaxes the necessary automated content realism and believability standards to "good enough", where good enough can be hu-

man readable but not necessarily context appropriate or grammatically correct. There are three primary methods envisioned for automated creation of document content, sampling from existing documents and statistical or algorithmic generation. The content generated from any of the three methods can then be incorporated into documents either via Document Content Redaction by Complete Substitution as described in Section 2.2.1 or through Automated Control of Applications for Document Creation as described in Section 2.3.

In general, we concentrate below on textual content creation, but similar methods are suitable for graphics and photographs.

2.2.1 Redaction of Existing Document Content

Redaction of Existing Document Content is analogous to Document Metadata Redaction from Section 2.1.1, but in document content the problem solutions are less well defined. There are two basic methods to redact privacy-related data from existing files, direct substitution of constructive information for privacy-related information without changing other parts of a document or simply replacing the entire content of the document with constructive content. The latter is easier and guarantees complete freedom from privacy-related information in the files on a constructive disk image.

Direct Privacy-related Information Redaction Unlike document metadata which has limited content per field and that content is semantically consistent with the field's descriptor, the rest of the document is under no such conventions or restrictions. Privacy-related information can appear anywhere in a multitude of forms. Not only is there directly identifiable personal privacy information such as names, addresses, phone numbers and financial account information, etc., but also indirect information which could be used for identification such as nicknames or depictions of unique events in specific locations which could be cross referenced with other publicly available information to generate likely identifications.

Regular expressions can capture many of the formated privacy-related information snippets and provide templates for creating constructive data, but there are many naming variations which regular expressions cannot distinguish because they differ from other strings semantically, not pattern-wise. With time and incorporation of techniques from computational linguistics direct redaction of privacy-related information such as names and locations may be accomplished with accuracies on par with human performance of the task. This however does not help to identify

and remove any indirect information as well a still failing to provide a *guarantee* that all privacy-related information has been removed. Thus direct redaction of privacy-related information, leaving other parts of the originating document untouched, is not recommended.

Document Content Redaction by Complete Substitution Removing all text from a document and substituting constructive content does provide the complete guarantee of a privacy-related information-free document. Making text substitutions in documents to accomplish this is straightforward when methods such as those in Section 2.3.3, Remote Control of Applications are used. The requirements for the alternate text can be specified in number of bytes to approximate the footprint of the original document and new text can either be statistically/algorithmically generated, or sampled from another document. A weakness of both generation and sampling is that the result, although human readable, is not contextually or semantically related to the original or other documents on the constructive disk image. This makes these methods less desirable for creation of files central to a specific desired scenario or scenario set. For files in scenario sets which must be highly realistic, content generation should be accomplished by human content providers. Regardless of the method used to create the replacement content, once it is available the remote application control can modify the original document by **Select All** and **Paste** the new content, then **Save As** onto the constructive disk image.

2.2.2 Document Content Creation from Scratch

Creating constructive documents from scratch avoids the challenges and difficulties of achieving guarantee-able results in removing privacy-related information via redaction schemes. The simplest method is human creation of document content. Document control is fine-grained and is highly controllable but speed which constrains the scale of the effort, cost and accuracy are factors which must be considered. Generation of text in a completely automated manner can be done and is commonly accomplished using statistics-based Hidden Markov Model chains, or algorithmically from a context-free grammar (CFG) ruleset. Both methods can produce grammatically correct text, but generally create semantically incoherent prose. A middle ground is using existing public domain documents which do not contain privacy-related information and sampling text. Sampling will provide a document where most of the content is grammatically and semantically correct, but may begin and end abruptly if sample boundaries are not coincident with the sentence endpoints in the sampled document. More specific discussions of each

method follows.

Human Generation For files in scenario sets which must be highly realistic, content generation should be accomplished by human content providers. This allows fine grained control over all aspects of the document content, formatting and indirect relationships to other document's content when they are part of the same scenario or scenario set. Large scale creation of privacy free document content by paid labor can be expensive so it is recommended that this technique be used when realism and/or inclusion of specific content is required, not for mass creation of documents which will comprise the drive's "background noise".

Statistical Text Generation There are several methods of generating text based on statistical analysis of a set of source document. Because a source document is used, it is prudent to use documents which do not contain directly identifiable privacy-related information. Choosing privacy-related information-free documents or redacting/substituting information such as addresses, phone numbers credit card info, SSNs etc., is required or some of the privacy-related information may be reconstituted in the resulting document. With a large source document library, arbitrary volumes of unique text can be generated.

Cut-up techniques are the least statistical and revolve around word re-arrangement, literally referring to cutting up the page and pasting the words in a new order. Source code for a version of a text cut-up algorithm written in Perl, Transcramble [6], is available online as shared code. Hidden Markov Model (HMM) chaining algorithms also take source documents, create simple statistical relationships based on locality and select text for inclusion in the new file based on these locality statistics. As each word is selected the newly resulting locality statistics of the target document are used to affect the randomized choices for the next word. There are several example HMM generators, with multiple programming language variations and available source code on the web. Dissociated Press is another related statistical technique that has features of both cut-up algorithms and HMMs. Dissociated Press can perform character level generation and create plausible compound words in addition to word level text generation. There is an implementation of Dissociated Press included in the Emacs text editor [2].

With an extended effort, a library of exemplar documents can be developed to serve as document type-specific sources for statistical text generators. This can be quite useful for enabling richer interactions in networks of Agent-based constructive users as discussed in Section 2.3.4.

Algorithmic Text Generation with Context-Free Grammars A famous example of CFG generation is the SCIgen project [8]. SCIgen generates random content academic papers in the domain of computer science and was used to generate a paper that was accepted to a conference. Without delving into any of the merits of that situation, SCIgen created a paper that at a glance, without actually "reading" it, looks as it it could have been written by a human. Reading a snippet here and there show grammatical correctness, although the semantic content begins to become suspect. This grammatical correctness and visually convincing appearance seems to indicate CFG may be a very good candidate for generating large volumes of text which can be incorporated into the "background noise" that surrounds the files central the the constructive drive image scenario set. There also exist CFG image generators which can be used to generate simple graphics for chart and figure inclusion into documents.

With an extended effort, additional CFG generators for many types of documents may be developed and can be quite useful for generating more varied types of documents and enabling richer interactions in networks of Agent-based constructive users as discussed in Section 2.3.4.

Document Sampling Sampling of existing human created documents can be used to generate content which upon inspection is obviously written by a human. If public domain documents, either by explicit license or government sourced, are selected as source material then the issues of privacy-related information is avoided. The major concern when selecting source material is that the document is appropriately licensed or released for this type of derivative work use. Freely downloaded but copyright/license protected documents should be avoided. There are numerous sources for appropriate material such as government reports, documentation from open source software programs, or published fictitious and non-fiction works distributed under licenses such as the GNU Free Documentation License or Creative Commons License. One aspect to be determined before the non-public domain documents are used as source material is whether licensing terms must appear internally in each file constructed via sampling or if a single licensing file accompanying the image will meet licensing terms for the entire image as a derivative work.

If a small number of source documents are being used to generate a large number of target document text samples, it is possible to sample across the same text from the same source. The probability of starting and ending the sample in *exactly* the same places is extremely small if proper randomization techniques are used. But if absolute guarantees of unique text samples are

required to ensure unique disk sector MD5 hash values, the newly created samples can be MD5 hashed, compared and in the extremely unlikely case of collisions, a sample may be obfuscated by performing random character replacement every 256 characters or less. The 256 character max is chosen to account for 2-byte unicode characters and a sector size of 512 bytes, if smaller sized sectors are used analogous choices may be made.

Using sampling techniques, several large documents can provide a near infinite number of unique coherently readable text samples ranging from sentence fragments to long documents. Like the other text generation methods, individual samples do not automatically yield complex document constructions such as tables of contents, cross-referencing or other indexing. For documents intended to provide the "background noise" on a constructive drive image this is not a significant limitation.

2.2.3 Within Application Document Creation

Creating actual files is the second step in the process of getting content onto the drive if any of the Section 2.2.2 methods are used to generate content. Using the native applications and filesystem in its normal running mode to create, save and maintain documents to the constructive drive image avoids all the negatives discussed below in Section 3.3.1 and ensures all application specific and filesystem content, as well as metadata, are correctly written from an application and filesystem standpoint.

The raw content which will eventually be incorporated into documents on the constructive drive can be created on a different machine than the computer being manipulated for generating the constructive image, or content can be created on a scratch drive within the same machine. We anticipate it will be simpler to avoid creation artifacts on the target constructive drive image if content creation is done on a separate machine and then accessed via network. The network accessed content can then be pasted into a document being manipulated on the target machine. The basic procedure to do so is fairly straightforward for a human operator: run the application to create, save and/or modify documents. This manipulation and/or creation of documents by a human operator can be quite slow, and if a large number of documents are required, monotonous. Automated application control can be accomplished to relieve monotony and increase both speed and accuracy, several methods for doing so are discussed in Section 2.3. Specifically, within application document creation with the content creation accomplished on a networked computer is compatible with the Remote Control of Applications methods discussed in Section 2.3.3.

2.3 Automated Control of Applications for Document Creation

There are several potential methods for automated control of applications: programmatic GUI control/operating system provided script driven methods, QA GUI testing software which may is generally not capable of remote control of a target computer, or virtual network computing (VNC) remote control of a target computer.

2.3.1 Programmatic GUI Control

There are open source and proprietary software packages available that assist with the creation of programs that can inspect applications written on top of object-oriented GUI frameworks at runtime and forward messages to the underlying framework as if it came via mouse or keyboard input. Examples of programmatic frameworks are the open source pyWinAuto which is a Python language package [4]; the freeware but proprietary software package AutoIt which provides a scripting interface to the underlying AutoIt engine doing the work [1]; Visual Basic for Applications provides GUI control interfaces for programmatic use within the Microsoft family of software products [7].

All three of these packages have a common usage limitation, they only support MS Windows applications. AutoIt and VisualBasic are further limited controlling to COM and .NET framework implemented applications. This may not be a terrible limitation as MS Office is essentially ubiquitous for basic document creation, but the ability to create documents and the associated metadata from any application is still inherently limited. pyWinAuto does not share the COM and .NET limitations, but the tradeoff is that it is very manually intensive to create new application automation scripts and programs. GUI component handles must be discovered and explicitly programmed into the pyWinAuto script via name strings. With no singular naming standard or convention, laboriously gathered information from earlier automated applications is of limited value since different programmers and programing teams can use a near limitless number of variations for naming the same components.

On operating systems other than Windows, Automator is a GUI scripting automation tool under Mac OS X which has replaced AppleScript. It works much as the above do, but has generally wide support across applications in OS X which are written in the Cocoa and Carbon APIs. Linux appears to be missing the necessary GUI consistency across its various distributions to enable developers to create a generalized tool. There are GUI testing tools based on X11 GUI testing packages which can automate some Linux applications, but no widespread programmatic

or script-driven Linux GUI automation solution

Another limitation in these all programmatic methods is that they must be resident on the machine that the application being manipulated is installed upon. Thus there will be artifacts left on a drive image from running the application. If the priority for an artifact-free image is high this method becomes difficult to use as image post-processing will be required, and there is risk that the post processing itself can leave some form of artifact even if indirect.

2.3.2 QA GUI Testing Software

There is a wide variety of software development quality assurance testing software available which incorporates automated GUI testing capabilities. The software packages investigated were each targeted at testing software developed with specific GUI APIs, not generalized solutions. It appears the the overall GUI product space is well covered in aggregate. All examined packages incorporated a scripting component compatible with the targeted API set enabling repeatable testing, but no software package featured general testing across a majority of available GUI development APIs. A more serious limitation of attempting to use these QA packages is that they are designed to run in the presence of the application source code. Access to the source is used by the testing engine to resolve GUI widget handles as the test script executes. Since we will not have access to the source of nearly all of the applications we will use to create a constructive disk image, QA GUI testing software is not a viable method to drive applications to create a disk image.

2.3.3 Remote Control of Applications

Remote application control can be accomplished using Virtual Network Computing (VNC) software. VNC is a means of graphically sharing computer desktops and allowing complete control of the client computer from a remote location. VNC uses a protocol termed RFB, the Remote Frame buffer, which operates virtually at the level of a user. We say at the level of the user because the server computer's frame buffer (what is streamed to the server's monitor with each vertical monitor refresh) is accessed directly for display on the (remotely controlling) client computer, and the client computer forwards simple mouse and keyboard commands back to the server computer. This gives VNC generality in program control, we are not hampered by matching development APIs and naming conventions.

VNC is also cross platform. There are client and server implementations on all three of the major personal computer platforms, Windows, OS X and Linux. Also, VNC clients on one

platform are able to control the target computer even if it is running a different operating system. This additional layer of generality reduces the level of effort needed to learn how to create new scripts for application control and document creation scripts across all three operating system platforms. VNC has a large number of implementations, both open source and proprietary, some of which come bundled with various add-on capabilities. The add-on capabilities of the VNC packages Eggplant [9] which is proprietary and VNCRobot [5] which is freeware, are scripting languages which can drive the VNC session in an automated fashion. Script driven VNC sessions can provide both generality and scalability when those scripts can access lists of external information such as constructive document content and target filenames. This makes a very powerful method for generating documents and associated metadata within the generating application. From examination of their respective websites, it appears Eggplant currently has this ability and that it is a future planned capability of VNCRobot.

VNC does require installation of the VNC server software on the machine that we are creating the constructive disk image on, so there are some artifacts in the image. Mitigating the introduction of VNC server software installation artifacts is that VNC is a very common protocol used for Help Desk software, for any drive scenario sets used for generating an image which is constructively part of some corporate or other large organization, the inclusion of VNC server software is not out of the ordinary. Many non-organizational users also install VNC server software to facilitate remote support by friends and relatives so it would not be considered entirely artificial for VNC server software to reside on the majority of potential disk images that can be created.

2.3.4 Future Agent-based Constructive Users

With future work it should be possible to create software agents to serve as constructive users and have networks of these agents exercising their constructive disk images via application remote control. The agent software would have the role of generating content that is as scenario applicable as the state of the art will allow and then scheduling a script which can use one of the remote control methods to actually execute the desired file/communications operations. Execution of those operations will embed the associated metadata and content into the generated files. These agents should operate from their own machine and use the networked remote control methods to preclude agent specific artifacts from becoming embedded in the target image. With a number of agent machines controlling their target machines, it should be possible to generate scenarios that exhibit simple social networking.

3 Filesystem

This section discusses the process of taking the documents generated from the previous section and writing them to the constructive drive image. This requires that the file be physically placed into sectors of the drive image and that the file directory structures be modified to reflect the addition of the file to the drive image. Most of the considerations for generating documents and document metadata from Section 2 are still valid, but we will cover some additional issues as well.

The filesystem contains its own metadata descriptions of files which need to be appropriately handled to maintain as much consistency as possible in the drive image's scenario set, both between other file metadata, and in conjunction with document metadata. As also seen with documents in Section 2.1, filesystems can contain "hidden" or "lost" information. This hidden information results from files or file fragments becoming disconnected from the filesystem directory through changes such as saving a file modification or deleting a file altogether. Normally, both of these operations eliminate the reference to the affected sectors without explicitly overwriting or destroying the information. There are filesystem operating preferences which can direct overwriting of the newly unreferenced sectors, but use of these options are not widespread as seen in [3].

Methods for building the filesystem on the drive image are analogous to those discussed for document creation: direct sector creation, within filesystem creation and filesystem redaction by substitution. The methods and relative merits of each will be discussed in Sections 3.3.1 thru 3.3.3, but first we will start with coverage of filesystem metadata and filesystem wear which are common concerns amongst all image creation methods.

3.1 Filesystem Metadata

The filesystem metadata encompasses all filesystem attributes beyond the directory tree structure. Many of these attributes are directly manipulated and maintained for filesystem directory display functionality, other attributes are not used directly by the filesystem but are maintained by the filesystem for other operating system functionalities such as indexed searches, custom icons, comments or filename label coloring. The metadata is stored and maintained in a filesystem specific manner which may be contained in the actual file, in an associated file fork, or both. The specifics of how these attributes are created and maintained are most germane to the direct sector creation method covered in Section 3.3.1 as the other methods take advantage of

the native filesystem to handle the metadata creation and storage location details.

As with document metadata, filesystem metadata plays a central role in defining the scenario set of the drive being examined. For files designated as being central to a specific scenario, consistency between the document metadata and file metadata can be critical. For many of the items that require this consistency, creating or manipulating the constructive file from within the appropriate application and allowing the application and filesystem to accomplish their respective tasks will suffice. The most problematic attribute to realistically affect appears to be time. Every log-able action in an operating system receives a time stamp, when constructing a constructive drive image in a naive manner, the time stamps for all operations, including file time metadata, will be compressed compared to a machine that has been in actual use. For single machine scenarios which do not rely on external inputs, manipulation of the system clock may suffice to correct this compression artifact as part of the drive image creation protocol. For scenarios where multiple users communicate and share messages or documents, timing synchronization becomes far more complex. Additionally, file metadata such as email headers can become quite difficult to generate in a realistic manner that properly encompasses time-dependent routing node choices as well as simple time stamps.

Another aspect to filesystem metadata is how to handle creation of shared documents that originate form a user other than the primary user of the target constructive drive image. Document-centric aspects of this were also discussed in Section 2.1. These shared documents should be constructed separately, on a different computer than the target machine. It will also be necessary to determine how many of these "other user" files to incorporate into the target image and how many users they should be from. These questions can be answered either by scenario-driven decisions or by statistical analysis of corpus drive images that are similar to the desired target image. A library of these shared documents can be generated and reused. Certain user-creator names may be designated as "generic users" who should not be considered part of a social network during drive analysis. Specific decisions related to these other-user related files should be made on a case-by-case basis using the objectives for the drive image creation as the main driver for what level of effort is necessary to create a constructive drive image with "good enough" realism for the intended purpose.

3.2 Filesystem Wear

Realistic disk images used for education and forensics tool testing must encompass not only the obvious file directory accessible information, but also the hidden information that is a result of filesystem wear. Wear is the largely result of sector use, disuse, and re-use over an extended period and is driven by day-to-day use of the machine to which the drive is connected. Software and firmware updates can also change filesystem operating details, these changes can affect how metadata is stored or details of which start sectors are selected when writing file fragments. Even reformatting a drive does not normally erase the actual physical information for files stored on the drive, it merely replaces the filesystem directory, orphaning all the content previously on the drive. Collectively, the wear results in sectors that are populated with data, but no longer accessible via filesystem directory entries.

These orphaned sectors are the operating domain of file-carving utilities and make up a very important component of a realistic constructive drive image. Having drive images created under controlled conditions can allow ground truth to be maintained, producing not only a drive image, but also the "answer key" necessary for results of forensic tools or student analysis to be compared against the actual drive contents. To ensure we maintain a good picture of the drive as it wears, it should be imaged and the sectors cataloged at frequent intervals during its construction. The optimal imaging frequency is yet to be determined, but it will depend on file deletion and modifications as well as how close to capacity the drive is during save operations. A relatively empty disk is much less likely experience an overwrite of orphaned user file sectors immediately compared to a disk that is substantially approaching its maximum capacity. Another factor to the optimal re-imaging frequency is behavior of the operating system as it autonomously manipulates files such as preferences, temp-files and ever growing logs, especially if the operating system has built in file defragmentation on-the-fly such as the HFS+ filesystem does in Mac OS X.

Imaging frequency can be extended if all user-initiated file operations are tracked and cataloged. We believe that writing each save operation to a scratch drive will provide enough cross-reference information to handle disambiguation and identification of orphaned sectors should multiple overwrites occur between imaging sessions. Once a drive image construction is complete the sequence of images and the scratch drive cross-reference can be used to catalog every sector of the constructive drive image conclusively.

Further investigation and development will be necessary for the drive imaging software so that it may invoked autonomously by the creation process and not leave artifacts on the constructed drive or require human intervention to physically put the drive in a different machine to image it. Enabling this form of automation will reduce bottlenecks in scaling the rate of constructive drive image creation. Specific solutions are anticipated to be operating system specific with the most commonality between Linux and Mac OS X. We anticipate that it may require a significant effort to make the automated imaging capability compatible with automated Windows constructive drive creation.

3.3 Generating the Filesystem

Filesystem metadata and attributes are normally generated or updated as documents are saved or modified. A few attributes and metadata fields are independently editable via Windows Properties, Mac OS X Get Info dialogs. The command line may also be used in certain instances to modify attributes and metadata but the majority of users will either let the filesystem/application pair handle the metadata or adjust it via the provided GUI interfaces.

3.3.1 Direct Sector Creation

Writing the target drive contents to the physical media directly, outside of the filesystem that is being simulated yields the most direct control of where each byte of information is written. The total control comes at a cost of coding a complete filesystem simulator. In other words, if we desire to end with an NTFS target disk image, we would need to perfectly implement the NTFS filesystem protocols in our write-control software. The level of effort to write and verify this is substantial on both a time and financial basis. Supposing such a filesystem simulator was built, any changes to the actual filesystem over time necessarily need to be mirrored in the simulator. That further burdens a simulator with long-term maintenance and testing costs. Additionally, for each new filesystem that we wish to be able generate creative disk images, we must develop the corresponding filesystem simulator. The above listed negatives far outweigh any perceived benefits gained from complete control of where data is written to the physical media.

3.3.2 Within Filesystem Creation

Creation of constructive drive images by working within the filesystem is straightforward. Conduct all file operations from within applications! Thus the discussions in Sections 2.1.3, 2.2.3 and 2.3 are germane. These within application file manipulations may be from within GUI-

based applications or even from the command line. The key is that the operating system uses the native filesystem to do the writing, relieving the burden of perfect filesystem simulation from the researchers or technicians in charge of generating the constructive drive images.

Of special interest when generating the constructive drive image is time. As noted in Section 3.1, the system clock will need to be routinely manipulated during creation of constructive drive images to mitigate the time compression inherent in generating an image artificially compared to a real world user exercising the computer over months or years. Some files may even need to be older then the constructive images start date if they are simulated as being migrated from a users previous computer or transferred from a different user altogether. It will be necessary to manipulate the system time not only during creation and manipulation of target disk image files, but it will also be necessary during the creation and/or modification of other-user files such as those discussed in Section 3.1.

3.3.3 Filesystem Redaction by Substitution

The basic concerns of filesystem redaction are the same as those for document metadata redaction discussed in Section 2.1.1. Additionally, true redaction by substitution requires the ability to do direct sector creation discussed in Section 3.3.1. We feel this is largely impractical for the reasons presented in that section. The next section proposes a hybrid method of redaction by substitution and within filesystem creation that shows promise.

3.3.4 Hybrid Within Filesystem Redaction

In this proposed hybrid method, a source drive image is used to determine what number and types of of files to create and the metadata fields that should be present in the new file written to the target drive. The source filesystem directory and metadata can be used to sort what system metadata-affecting settings should be active for the writing of a particular file set, and what time-stamps should be used to manipulate the system clock during target drive creation. Similar determinations should also be made on per-application basis for application specific metadata-affecting settings. These filesystem and application settings can be used to create a plan for exactly what the constructive drive files and metadata contents should look like, including the redaction substitutions where they are needed. Decisions on file content may or may or may not be affected by this planning process dependent on the level of content/metadata consistency required.

It should be possible to specify the resulting plan in such a way that it acts as a dynamic con-

figuration control for the target constructive disk image process. Well-specified document creation/manipulation configuration files and corresponding content creation configuration files should dovetail into enabling a fully automated target disk image generation utility. It is further recommended that this utility use the remote control method to drive the creation of the target disk image. Remote control methods may or may not be necessary for the pre-processing steps, direct programmatic methods may be more appropriate as there is no fear of generating artifacts on the constructive image at that stage.

One aspect that the hybrid method does not address via examination of the source drive's filesystem directory is orphaned sectors. To handle this we can examine the source image for sector orphan volume and use that figure as the desired volume of sector orphaning to include on the target image. Once this is determined, file manipulation operations specifically designed to create orphan sectors can be inserted into the overall drive creation plan. As the automated target disk image generation utility matures, additional statistically based analytical tools can be incorporated to allow further fine tuning of target image file content and orphan sector content.

THIS PAGE INTENTIONALLY LEFT BLANK

References

[1] Jonathan Bennett. AutoIt v3 - automate and script windows tasks, 2009. URL http://www.autoitscript.com/autoit3/.

[2] Free Software Foundation GNU Emacs contributors. Dissociated Press - GNU Emacs manual, 2007. URL http://www.gnu.org/software/emacs/manual/html_node/emacs/Dissociated-Press.html.

[3] Simson Garfinkel and Abhi Shelat. Remembrance of data passed. *IEEE Security and Privacy*, January 2002.

[4] Mark Mc Mahon. pyWinAuto, 2006. URL http://pywinauto.openqa.org/.

[5] Robert Pes. VNCRobot - automated testing tool based on VNC technology, 2009. URL http://www.vncrobot.com/index.html.

[6] Briac Pilpré. Transcramble - random text generator, 2001. URL http://www.perlmonks.org/index.pl?node_id=94856.

[7] Ed Robinson. Automating the creation of data-rich business documents with Word 2007 and Visual Basic 2005, March 2007. URL http://msdn.microsoft.com/en-us/library/bb407305(vs.80).aspx.

[8] Jeremy Stribling, Max Krohn, and Dan Aguayo. SCIgen - an automatic CS paper generator, 2005. URL http://pdos.csail.mit.edu/scigen/.

[9] Testplant Ltd. QA Automation — Eggplant Functional Tester — Redstone Software, 2008. URL http://www.testplant.com/products/.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Barbara Guttman
   National Institute of Standards and Technology
   100 Bureau Drive
   Gaithersburg, MD 20899-8970

4. Dr. Simson Garfinkel
   Naval Postgraduate School
   Monterey, California

5. Loren Peitso
   Naval Postgraduate School
   Monterey, California